Introduction

Quality assessment is an essential step in the wine making process as it helps ensure consistency and consumer satisfaction. The quality of wine has historically been evaluated by wine makers and sommeliers, who leverage their training and expertise to determine if a product meets acceptable standards (1). While there have been advancements in the detection of the physicochemical properties of wine and how they relate to flavour (2), quality assessment remains a subjective metric.

Here we ask if wine quality can be predicted using machine learning. Our focus is to create an interpretable linear regression model trained on physicochemical wine data to perform this task. The utility of such a model is two-fold as it can be used as a tool to help less experienced wine makers and sommeliers get a sense of the quality of a given wine. Furthermore, the model would help to provide a more objective framework for wine assessment through interpretation of its coefficients. To achieve this, we will train our model on data related to vinho verde white wines from the Minho region of Portugal which contains the physicochemical and sensory data of 4898 wines (3).

In [11]:

```python
import pandas as pd
import altair_ally as aly
import altair as alt
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, RidgeCV
from sklearn.metrics import root_mean_squared_error, r2_score
```

In [12]:

```python
aly.alt.data_transformers.enable('vegafusion')
```

Out[12]:

```
DataTransformerRegistry.enable('vegafusion')
```

In [13]:

```python
origin_df = pd.read_csv('data/winequality-white.csv', sep=';', encoding='utf-8')
```

In [14]:

```python
# looking at head and tail of data
origin_df
```

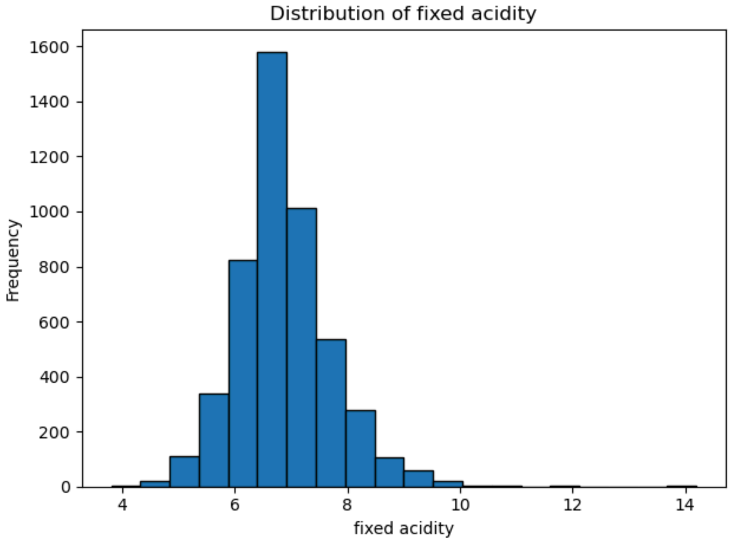
Out[14]:

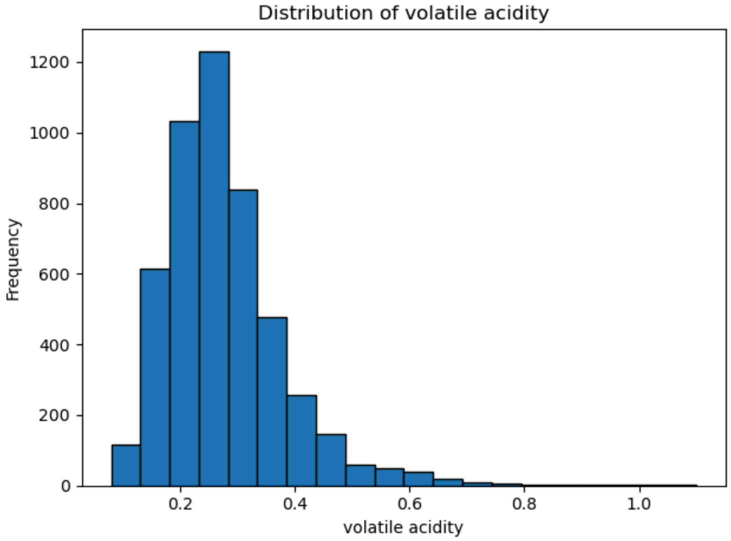
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
...	...	...	...	...	...	...	...	...	...	...	...	...
4893	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6
4894	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5
4895	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6
4896	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7
4897	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6

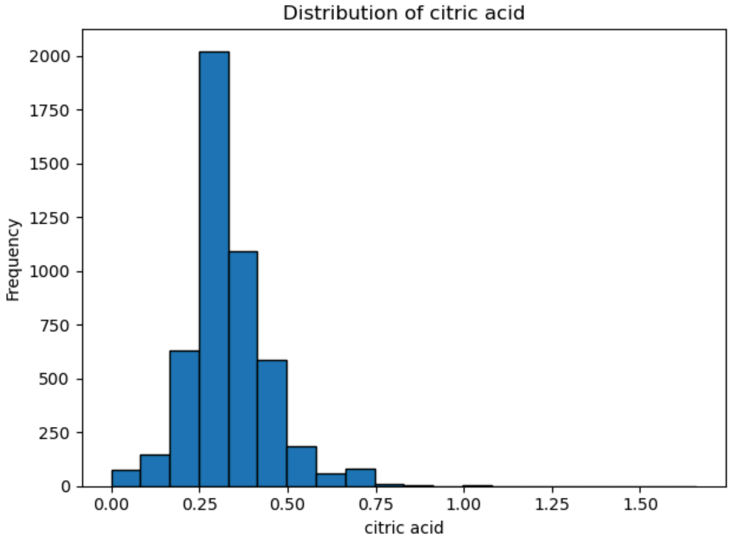
4898 rows × 12 columns

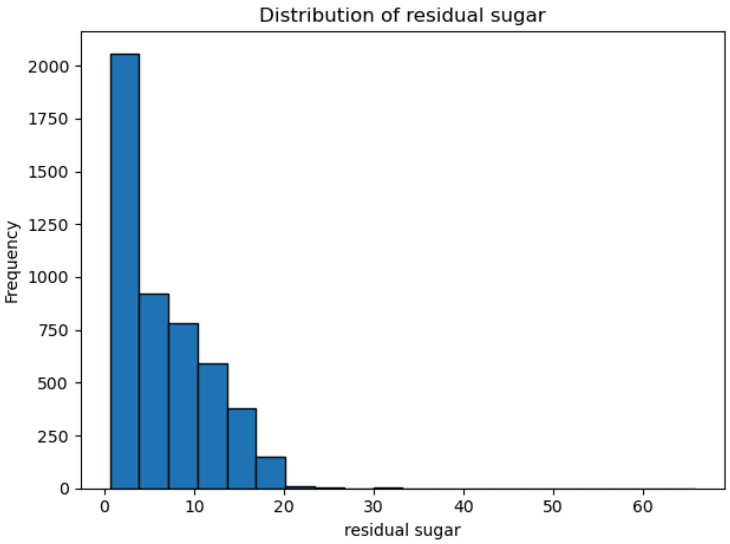
In [15]:

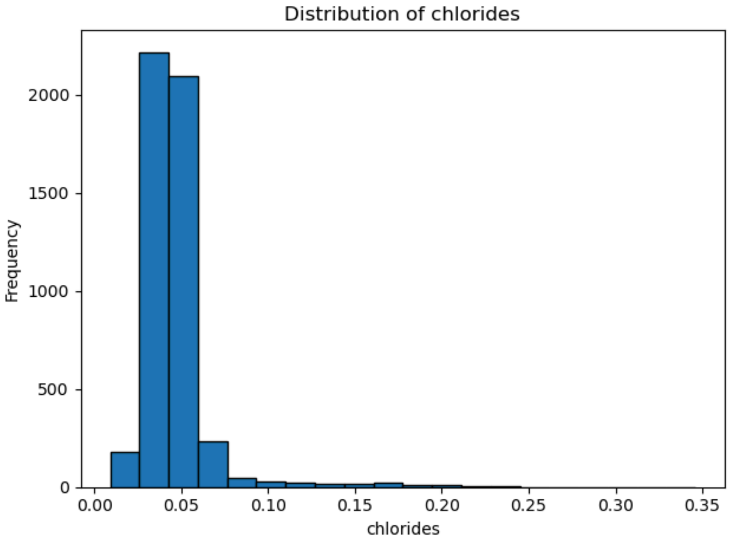
```python
# plotting a bar graph for each variable
for feat in origin_df.columns.tolist():
    plt.hist(origin_df[feat], bins = 20, edgecolor='black')
    plt.xlabel(feat)
    plt.ylabel('Frequency')
    plt.title(f'Distribution of {feat}')
    plt.tight_layout()
    plt.show()
```

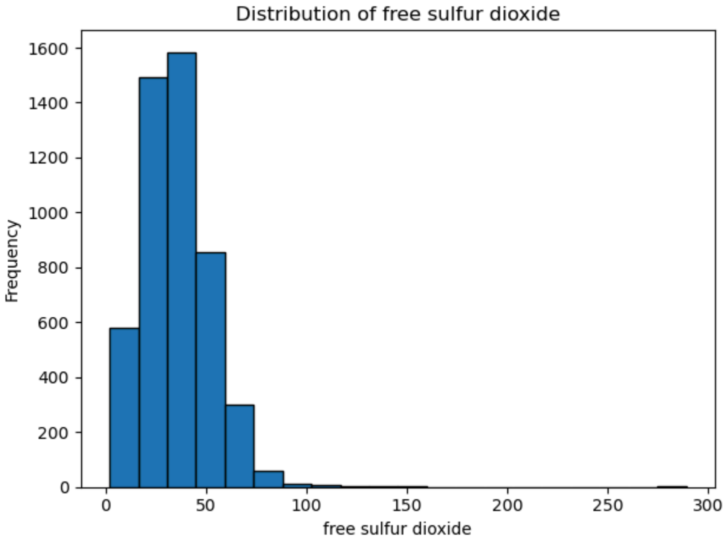

Distribution of fixed acidity
Frequency
0
200
400
600
800
1000
1200
1400
1600
4
6
8
10
12
14
fixed acidity

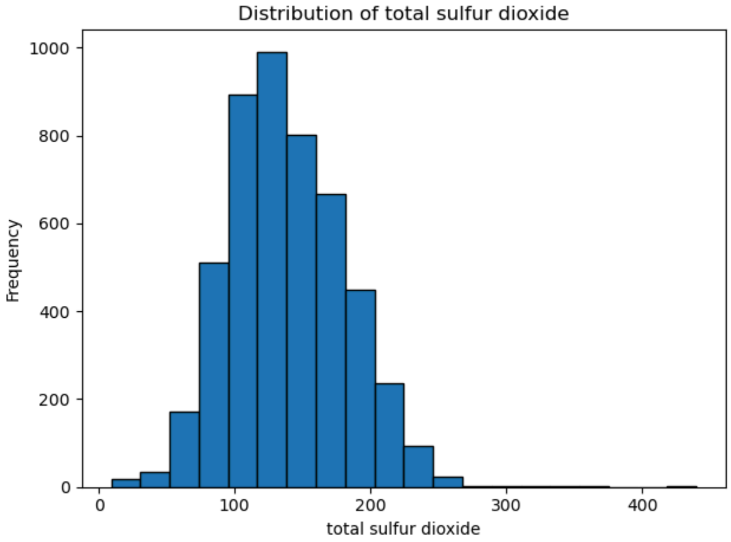

Distribution of volatile acidity
Frequency
0
200
400
600
800
1000
1200
0.2
0.4
0.6
0.8
1.0
volatile acidity

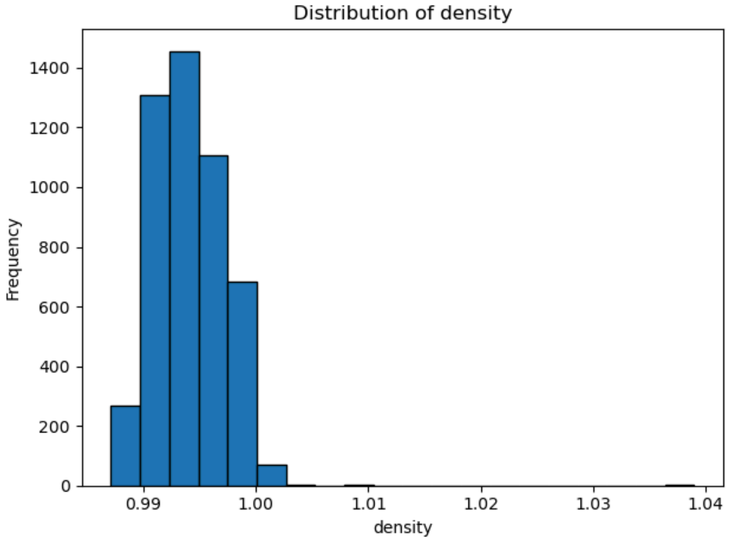

Distribution of citric acid
Frequency
0
250
500
750
1000
1250
1500
1750
2000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
citric acid

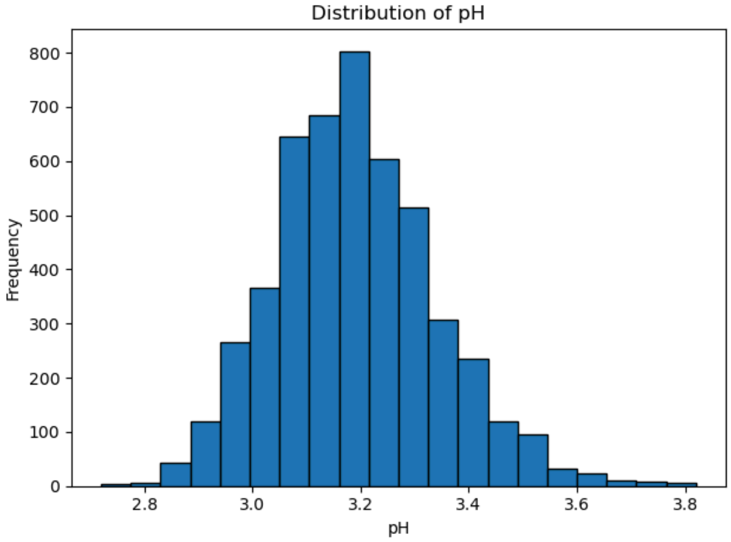

Distribution of residual sugar
Frequency
0
250
500
750
1000
1250
1500
1750
2000
0
10
20
30
40
50
60
residual sugar

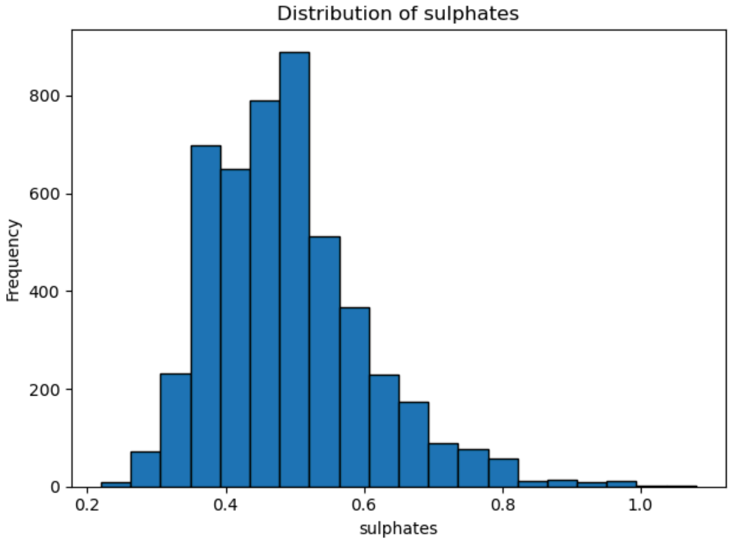

Distribution of chlorides
Frequency
0
500
1000
1500
2000
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
chlorides

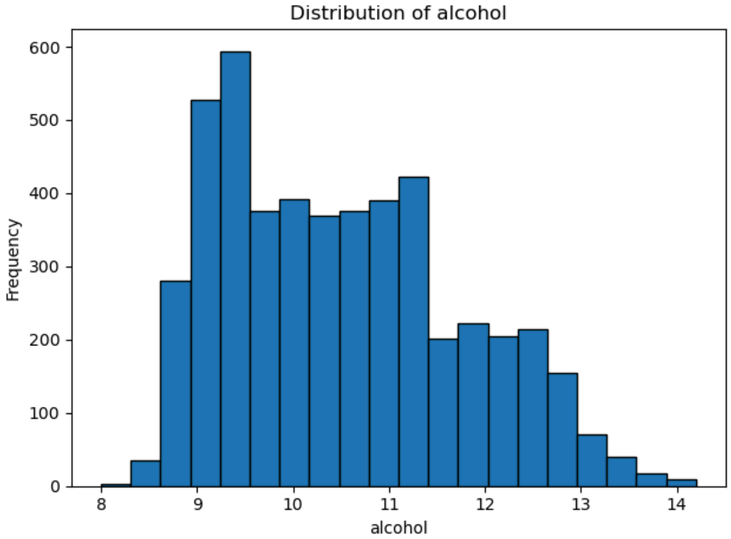

Distribution of free sulfur dioxide
Frequency
0
200
400
600
800
1000
1200
1400
1600
0
50
100
150
200
250
300
free sulfur dioxide

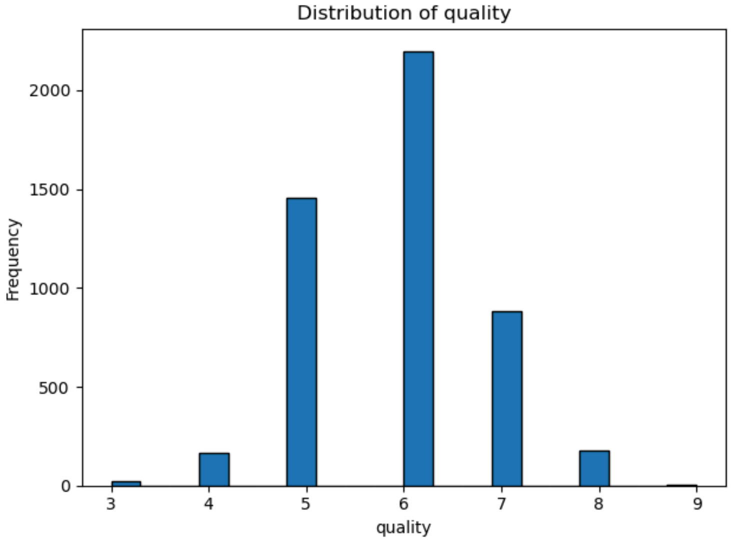

Distribution of total sulfur dioxide
Frequency
0
200
400
600
800
1000
0
100
200
300
400
total sulfur dioxide


Distribution of density
Frequency
0
200
400
600
800
1000
1200
1400
0.99
1.00
1.01
1.02
1.03
1.04
density


Distribution of pH
Frequency
0
100
200
300
400
500
600
700
800
2.8
3.0
3.2
3.4
3.6
3.8
pH





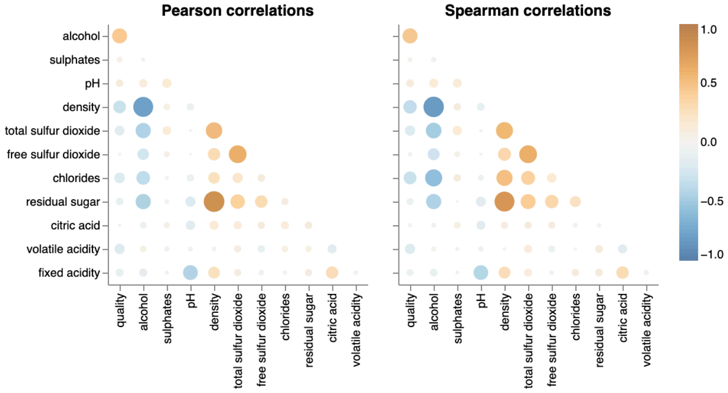


By taking a look at the individual distributions of the variables, we can see that the majority of them are approximately normal with some degree of right-skew. This tells us that most of the values tend to cluster around the average, possibly an industry standard, with the more extreme values tending to fall above the average rather than below it. The distributions of residual sugar and alcohol content break this pattern as the residual sugar plot follows a more exponential pattern, and the distribution of alcohol content is still approximately normal but also shows a tendency towards uniformity.

```
In [16]: aly.corr(origin_df)
```

```
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
c:\Users\7luck\miniforge3\envs\dsci573\Lib\site-packages\altair\utils\data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.Data
Frame'>` to `is_pandas_dataframe`.

Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?
  return _is_pandas_dataframe(obj) or isinstance(
```

Out[16]:



By taking a look at the Pearson and Spearman correlations, we can get a sense of what variables will be more informative about the predicted quality and which variables may be collinear. Based on the plot, a higher alcohol level is associated with a higher quality rating, while a higher density and chloride value is associated with a lower quality rating. Also, the plot suggests a strong linear correlation between multiple variables such as density and residual sugar and density and alcohol.

In [17]:

```python
# separate the response and explanatory variables
X = origin_df.drop(columns=["quality"])
y = origin_df["quality"]
```

For exploratory purposes, we first create an ordinary least squares linear regression model including all predictors:

In [18]:

```python
import statsmodels.formula.api as smf
```

```
model = smf.ols("y ~ X", data=origin_df)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.282
Model:                            OLS   Adj. R-squared:                  0.280
Method:                 Least Squares   F-statistic:                     174.3
Date:                Sat, 22 Nov 2025   Prob (F-statistic):               0.00
Time:                        12:26:25   Log-Likelihood:                -5543.7
No. Observations:                4898   AIC:                         1.111e+04
Df Residuals:                    4886   BIC:                         1.119e+04
Df Model:                          11                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    150.1928     18.804      7.987      0.000     113.328     187.057
X[0]           0.0655      0.021      3.139      0.002       0.025       0.106
X[1]          -1.8632      0.114    -16.373      0.000      -2.086      -1.640
X[2]           0.0221      0.096      0.231      0.818      -0.166       0.210
X[3]           0.0815      0.008     10.825      0.000       0.067       0.096
X[4]          -0.2473      0.547     -0.452      0.651      -1.319       0.824
X[5]           0.0037      0.001      4.422      0.000       0.002       0.005
X[6]          -0.0003      0.000     -0.756      0.450      -0.001       0.000
X[7]        -150.2842     19.075     -7.879      0.000    -187.679    -112.890
X[8]           0.6863      0.105      6.513      0.000       0.480       0.893
X[9]           0.6315      0.100      6.291      0.000       0.435       0.828
X[10]          0.1935      0.024      7.988      0.000       0.146       0.241
==============================================================================
Omnibus:                      114.161   Durbin-Watson:                   1.621
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              251.637
Skew:                           0.073   Prob(JB):                     2.28e-55
Kurtosis:                       4.101   Cond. No.                     3.74e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.74e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

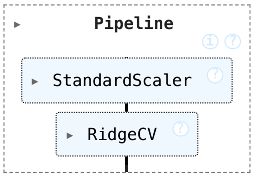
This model has an R-squared value of 0.282, meaning that the model explains roughly 28% of variability in the data. The R-squared value is close to that of the adjusted R-squared, suggesting most predictors contribute explanatory power to the model.
The p-value of the F-statistics is small, indicating that at least one predictor has a statistically significant assocation with the response.
The output reveals that the condition number of the model is large, possibly indicating strong multicollinearity. Thus, we will opt for a ridge regression to induce an L2 penalty on correlated variables.

In [19]:
```
numeric_features = X.columns
numeric_transformer = StandardScaler()
```

In [20]:
```
# create preprocessing pipeline
model = Pipeline(
    steps=[
        ("scaler", StandardScaler()),
        ("regressor", RidgeCV())
    ]
)
```

In [21]:
```
# split data into testing and training sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=123
)

# fit the model on the training data
model.fit(X_train, y_train)
```

Out[21]:



In [22]:
```
# use the model to predict on the testing data
y_pred = model.predict(X_test)

# evaluate RMSE and R-squared values
rmse = root_mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse:.3f}")
print(f"R^2: {r2:.3f}")
```

```
RMSE: 0.734
R^2: 0.300
```

In [23]:
```
linreg = model.named_steps["regressor"]
coef_df = pd.DataFrame(
    {"feature": X.columns, "coefficient": linreg.coef_}
).sort_values("coefficient", ascending=False)
```

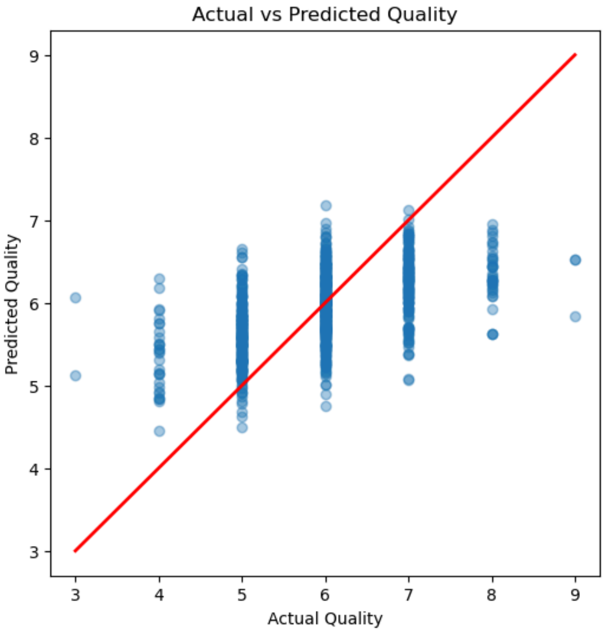
```
# display the model's coefficients
coef_df
```

Out[23]:

	feature	coefficient
3	residual sugar	0.378340
10	alcohol	0.260201
8	pH	0.089316
9	sulphates	0.076358
5	free sulfur dioxide	0.067991
0	fixed acidity	0.029480
4	chlorides	-0.002330
2	citric acid	-0.003946
6	total sulfur dioxide	-0.024003
1	volatile acidity	-0.187054
7	density	-0.385739

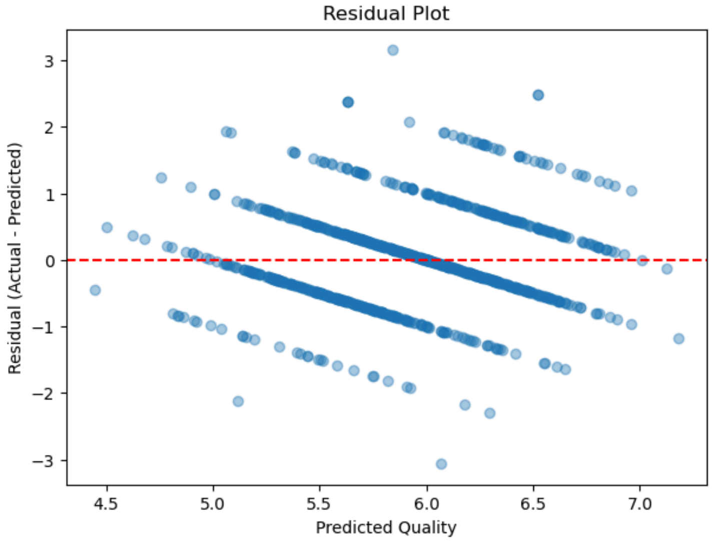
In [24]:

```
y_pred = model.predict(X_test)

plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred, alpha=0.4)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()],
         linewidth=2, color='red')
plt.xlabel("Actual Quality")
plt.ylabel("Predicted Quality")
plt.title("Actual vs Predicted Quality")
plt.show()
```



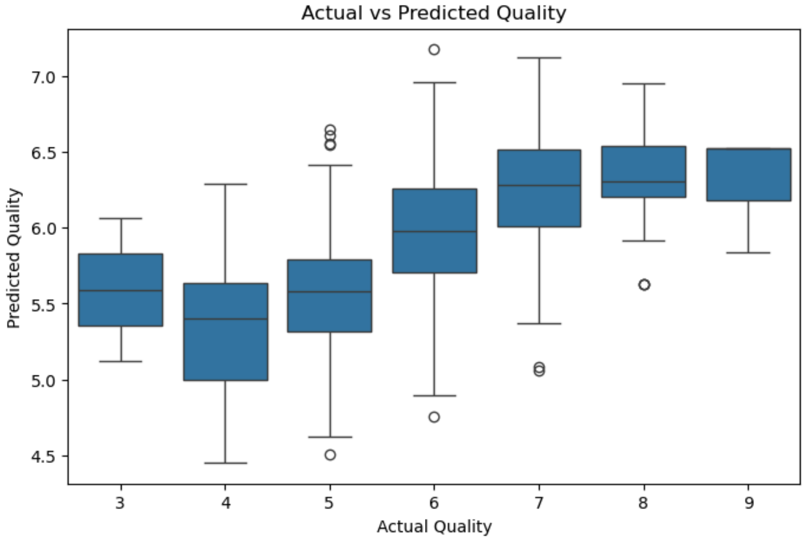
In [25]:

```
residuals = y_test - y_pred

plt.figure(figsize=(7,5))
plt.scatter(y_pred, residuals, alpha=0.4)
plt.axhline(0, color="red", linestyle="--")
plt.xlabel("Predicted Quality")
plt.ylabel("Residual (Actual - Predicted)")
plt.title("Residual Plot")
plt.show()
```



In [27]:

```
import seaborn as sns
import matplotlib.pyplot as plt

df_viz = pd.DataFrame({
    "Actual Quality": y_test,
    "Predicted Quality": y_pred
})

plt.figure(figsize=(8,5))
sns.boxplot(x="Actual Quality", y="Predicted Quality", data=df_viz)
plt.title("Actual vs Predicted Quality")
plt.show()
```



Discussion

When we look at the plot for actual versus predicted quality, we can see that our model had a tendency to predict the middling quality ratings, suggesting the model was biased by the high frequency of quality ratings of 5-7 in our data. While our ridge regression model did tend to predict higher quality values when the actual quality value was higher, the average prediction stayed between ~5 and 6.5 for all actual quality values.

After running our ridge regression analysis, the variables that the model found most informative for predicting the quality rating were the residual sugar and alcohol content, with a higher value of either being correlated on average with a higher predicted quality rating. Intuitively this makes sense, as wines with a higher alcohol content are considered to have more body and a richer taste, but there is also a tradeoff between alcohol content and residual sugar (4). The tradeoff was hinted at in our earlier EDA, as a higher alcohol content was associated with a lower residual sugar content. The alcohol content in wines is derived from a longer fermentation process, but this fermentation also consumes more of the sugar, so it's difficult to attain high levels of both.

These findings indicate that although taste is subjective, tasters have a tendency to prefer white wines that maximize the residual sugar and alcohol content, resulting in a full-bodied wine that is not too dry. Going forward, this could prompt winemakers to experiment with grapes with a higher sugar content and fermentation techniques that try to maximize the alcohol content while minimizing the sugar consumed during the process. This also leads us to questions about how aware sommeliers are of their preferences while judging the quality of wine and how closely the quality rating matches the average wine consumer's preferences. Do sommeliers just tend to enjoy sweeter, full-bodied wines, or is that their personal opinion due to their experience? More research could be done into this topic.

References

1. Langstaff SA. Sensory quality control in the wine industry. In: Sensory Analysis for Food and Beverage Quality Control. Woodhead Publishing; 2010. p. 236–61. https://doi.org/10.1533/9781845699512.3.236

2. Polášková P, Herszage J, Ebeler S. Wine flavor: chemistry in a glass. Chemical Soc Rev. 2008 Aug 12;37(11):2478–89. https://doi.org/10.1039/b714455p

3. Cortez P, Cerdeira A, Almeida F, Matos T, Reis J. Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst. 2009 Nov;47(4):547–53. https://doi.org/10.1016/j.dss.2009.05.016

4. Copestake N. How Much Alcohol is in Wine? A Complete Guide [Internet]. Coravin US. 2025 [cited 2025 Nov 22]. Available from: https://www.coravin.ca/blogs/community/wine-101-how-much-alcohol-is-in-wine